

# **NATURAL** **ENGINEER**

---

Managing Change

# Contents

<b>Managing Change with Natural Engineer</b>	<b>4</b>
<b>Natural Engineer – The Foundation</b>	<b>5</b>
The Basic Tool	
Flexible Configuration	
Complete Inventory of Application Code	
Impact Analysis	
Standard Impact Analyses	
Automatic Modification	
Audit Log	
Construct and Predict Case Support	
Unattended Operation	
<b>Increasing Maintenance Productivity</b>	<b>8</b>
Exploiting the Power of the PC	
Application of Coding Standards	
Integrated Structure Diagramming	
More Automated Approach	
Complex Program Code	
<b>Enhanced Documentation</b>	<b>10</b>
Comprehensive Reporting	
Code Beautification	
Links to Third-Party Flow Charting Software	
<b>Natural Version Migration</b>	<b>12</b>
Assessing the Impact of Migration	
Automating the Modification Process	
Unique Differences	
Supported Migration Paths	
<b>Application Restructuring</b>	<b>13</b>
Exploiting the Repository	
Building Program Objects	
Standard Restructuring Options	
Isolating Database I/O Operations	
Isolating User Presentation Operations	
<b>Application Performance Analysis</b>	<b>15</b>
Finding and Solving Performance Problems	
Measuring Program Complexity	
Performance Impact Analysis	
Automatic Modification	
<b>Additional Functions</b>	<b>16</b>
Compare Function	
Internationalization Support	
<b>Code Certification – A Critical Task</b>	<b>17</b>
False Economies	
Two Powerful, Cost-Effective Testing Tools	
The Benefits of an Integrated Toolset	
GenAda Testing with Real Production Data	
Using GenAda	
GenTest Automating the Testing Process	
Using GenTest	
<b>The Future</b>	<b>20</b>
Leveraging Repository Information.	
A Consistent Architecture	
Web-Enabling Natural Applications	
<b>Technical Environment</b>	<b>22</b>

# Managing Change with Natural Engineer

Change is here to stay. New technology, new business practices, and new requirements such as support for the euro, frequently force enterprises to make modifications to production applications. Ongoing software maintenance has become a fixed item in the IT budget, and often absorbs more funding than the development of new systems.

The ability of corporate IT to react to change is a critical business issue. For example, the cost savings expected from an acquisition or merger may depend on how quickly common IT systems can be implemented. Recent major re-engineering projects, however, particularly Year 2000 compliancy, showed that many organizations are ill-prepared to meet challenges of this kind. Not only was it extremely difficult to estimate the impact of required changes, but there was frequently no adequately documented application source code to work with.

Natural Engineer from Software AG can help you overcome many of the obstacles to change. It is a multi-purpose, multi-functional maintenance and re-engineering tool for Natural applications that comprises a powerful basic product with a range of options for special purposes, such as Natural version migration or application restructuring.

Natural Engineer enables you to respond faster to technological developments like the Internet, while fully leveraging prior investment in Natural applications. It offers you an easier route to the world of e-business, to enterprise-wide and cross-platform systems integration, or simply to better utilization of your development resources.

## **With Natural Engineer, you can:**

- Restructure Natural applications more easily to meet new challenges
- Migrate more quickly to a different version or platform
- Enhance software maintenance productivity
- Quantify the cost of major maintenance projects
- Assess the impact of proposed changes across all applications
- Manage maintenance and re-engineering projects more effectively.

This brochure provides a high-level introduction to these functions and describes how Natural Engineer enables IT management to establish better control of all aspects of application change and associated costs.



Change can result from legislation, mergers and acquisitions, or technology developments

# Natural Engineer – The Foundation

**Natural Engineer's powerful, versatile core technology supports a wide range of Natural code maintenance and re-engineering tasks. This is achieved by creating a complete inventory of application code in the Natural Engineer repository.**

## THE BASIC TOOL

Natural Engineer's core technology comprises a repository, a search engine, automatic modification facilities and auditing support. All additional functions, such as extended documentation are based on this foundation.

A simple analogy would be a domestic power tool with a series of optional, "snap-in" accessories. As with the domestic tool, new functions, to meet tasks not yet defined, will be developed in line with the standard design architecture. This means that new capabilities can be used with the core technology directly, eliminating the need for a new product or conversion process.



## FLEXIBLE CONFIGURATION

Natural Engineer is a Natural application and utilizes Adabas for its repository. As a result, it runs on a variety of platforms, including OS390, VSE and BS2000 host environments and with Windows NT on PCs.

This operational flexibility means that customers can use Natural Engineer in a totally mainframe environment, or in a standalone PC configuration, or in combinations of the two.

## A COMPLETE INVENTORY OF APPLICATION CODE

The repository - a single, comprehensive source of all application information in the enterprise - is the basis of all Natural Engineer functions.

The Extract and Load feature retrieves source code from operational libraries and loads relevant information into the repository. A powerful parser verifies the existence of every object referenced, and identifies redundant and missing objects. This ensures a complete inventory of all application code.

Comprehensive reporting facilities cover the entire process. Where objects are flexibly linked, for instance when a sub-program is called using a variable with the sub-program name, a feature known as Soft Links allows the user to specify a list of valid object names for the call. The use of Soft Links means that the integrity of the cross-reference information is always maintained.

The Extract and Load processes can be run separately, if required, and for large volumes of code, the repository can be built up over time with a series of incremental load executions.

Even the initial step of creating the Natural Engineer repository immediately provides a number of benefits for customers. An accurate, fully-documented description of all current application code is available with complete, online cross-reference data. Also, the powerful Natural Engineer parser adds value to the basic program information stored in the repository, facilitating later impact analysis and modification.

Natural Engineer caters for the automatic re-synchronization of the repository data after code modification. Based on date/time stamp information in saved Natural source code, a re-synchronization process can be initiated at user-defined intervals. The information stored in the repository is then updated automatically based on differences detected in the date/time stamp.

## IMPACT ANALYSIS

The powerful Natural Engineer search engine enables the user to quickly locate specific fields, files, statement types, etc., and report their frequency and location across all applications. It tracks impact chains across program and application boundaries to ensure that all potential impacts resulting from a required change are identified.

Natural Engineer Impact Analysis can generate reports at a summary, object and detail level, and impacted source code is always highlighted. More than 30 standard reports are available, allowing users to retrieve precisely the required details and reducing redundant information to the absolute minimum.



The tool allows detailed search criteria for a particular impact analysis to be created, and refined, very flexibly. A full range of partial and wild card functions can be added to the search criteria. Results from a specific impact analysis can be stored under different version numbers. Incremental Impact allows these results to be refined with additional criteria to produce just the information set required by the user. Incremental Impact can be carried out on the whole application, previously impacted objects or previously impacted statements. This gives maximum flexibility in precise identification of impacted code which could be later the subject of modification.

The comprehensive impact analysis functions of Natural Engineer give a full picture of what needs to be modified, across all applications and objects. This is essential to allow accurate estimation of the total impact and cost of a proposed change.

A key benefit of tool-based impact analysis is the elimination of missed impacts. This is often a problem in re-engineering projects involving complex code or high volumes when manual techniques or simple code scanning products are used. The cost of missed impacts can be significant if they are not detected until late in the test phase, or after the application has been returned to production.

#### **STANDARD IMPACT ANALYSES**

Natural Engineer provides a number of standard Impact Analyses with pre-defined search criteria. For example, MVS/NAT22TO31 includes the search criteria necessary to identify the impacted code when a user is planning a conversion from Natural Version 2.2 to Natural Version 3.1.

Other standard analyses include criteria for:

- Application restructuring with the Object Builder feature
- Identifying Natural formats, constructs etc that could have a negative impact on performance
- Identifying candidate text and fields for internationalization

These options are described in more detail later in this document.

#### **AUTOMATIC MODIFICATION**

Natural Engineer makes it easy to apply individual changes and pre-coded solutions automatically. The program modification facilities, including a flexible replace mechanism, are simple to use and increase developer productivity significantly.

After modification, Natural Engineer places all modified code into a user-defined, or default, Modification Library.

The automated approach to modification provided by Natural Engineer eliminates the need to make repetitive manual changes, a process that can be error-prone. It also reduces implementation time, so scarce development expertise can concentrate on areas requiring a unique, manual solution.

#### **AUDIT LOG**

All changes made to the program information stored in the repository are logged, with full details of what was changed, by whom and when. This provides a comprehensive audit trail that will meet the requirements of EDP auditors. Changes can be reviewed online or as hardcopy reports.

The standard change reports provided cover a wide the range of typical user audit requests. This makes it easy to answer requirements such as: "Show all changes made to the payroll system last year," or, "Show all changes made by developer x during the past 6 months."

As well as meeting auditing requirements, the change logging function of Natural Engineer can be used to give project managers the detailed information necessary to effectively oversee re-engineering and maintenance projects.

## **CONSTRUCT AND PREDICT CASE SUPPORT**

Natural Engineer was designed to support the requirements of user-developed Natural code. There are, however, facilities to support the use of Natural generators, i.e. Construct and Predict Case.

Natural Engineer “understands” and processes code produced by Natural Construct and Predict Case.

There are Construct-specific reports produced by all Natural Engineer processes and the tool differentiates between Construct-generated code and code in user exits. During the Modification phase Natural Engineer will only allow changes to the user exit code, not the Construct-generated code. If the user has modified the Construct model code, and it is therefore marked as no longer able to be regenerated by Construct, Natural Engineer treats the code as a normal Natural object.

Similarly, for Predict Case users, Natural Engineer produces specific reports relating identified impacts to Predict Case System Components.

## **UNATTENDED OPERATION**

An option is provided to allow the user to manage and execute the Natural Engineer Scheduler for unattended operation of the Environment Set-Up, Impact Analysis and Modification functions.

With the Natural Engineer Scheduler the user is able to select required functions for execution and specify a start date and time. The Scheduler takes control of the Natural Engineer session and provides a dialog that controls starting and stopping of the Scheduler’s execution.

The benefit of this feature is that long-running impact analyses or load processes can be scheduled for overnight execution when PC equipment is little utilized.

# Increasing Maintenance Productivity

**Natural Engineer offers several features that speed up and simplify the software maintenance process, cutting costs and releasing scarce resources for the development of new applications.**

## EXPLOITING THE POWER OF THE PC

Although Natural Engineer functionality can be used in both host and PC environments, the key productivity boost for using a highly-sophisticated tool such as Natural Engineer comes from using it on the PC where the additional, graphical features and local processing power can be used to maximum effect.

The use of the PC promotes a completely different approach to the maintenance process at the typical user site. Earlier methods for assessing the impact of required changes to an application system depended on batch processes and the use of limited scanning and editing tools.

In contrast, Natural Engineer on the PC promotes an incremental approach, allowing the user to "drill down" through application hierarchies and objects to identify the precise location and reason for a particular impact.

## APPLICATION OF CODING STANDARDS

The benefits of adopting strict standards for the design and coding of application systems are well known. The objective is to display application code with a consistent appearance and use of constructs offered by the programming language. This ensures that even a developer who is new to the application can quickly understand the logic and processes used.

However, although standards manuals and similar documents abound, it is often difficult to enforce standards rigorously across the whole organization.

Natural Engineer solves this problem once and for all by applying standards definitions to application source code automatically.

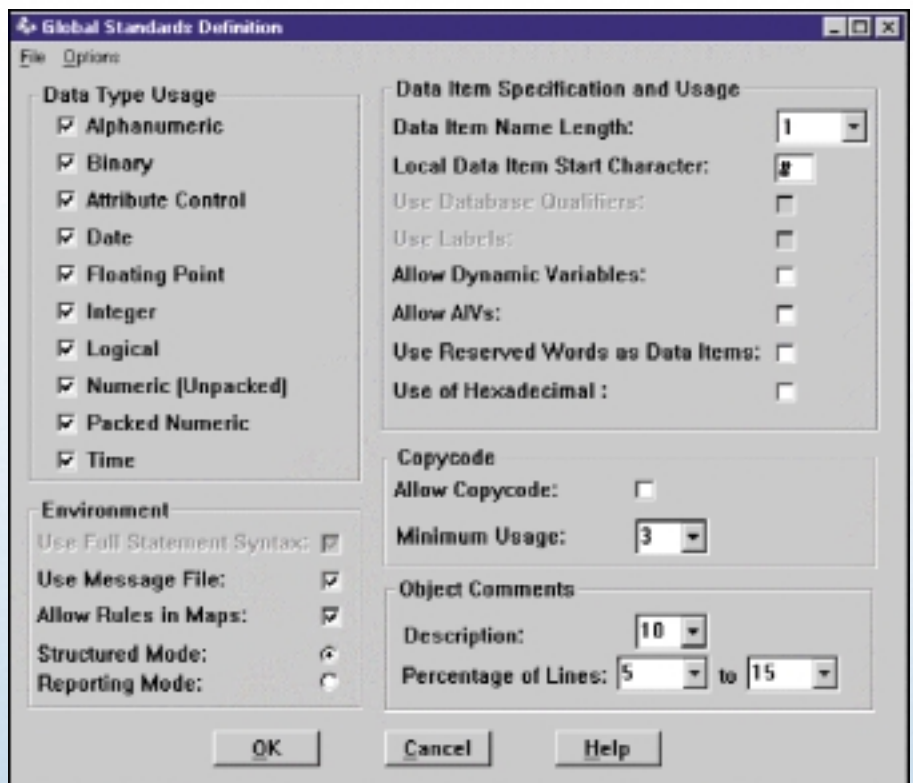
Typical specifications could include:

- Controls on program syntax
- Specification of minimum percentages of comments statements
- Fully-qualified database field names
- Use of reserved words
- Rules in maps

## INTEGRATED STRUCTURE DIAGRAMMING

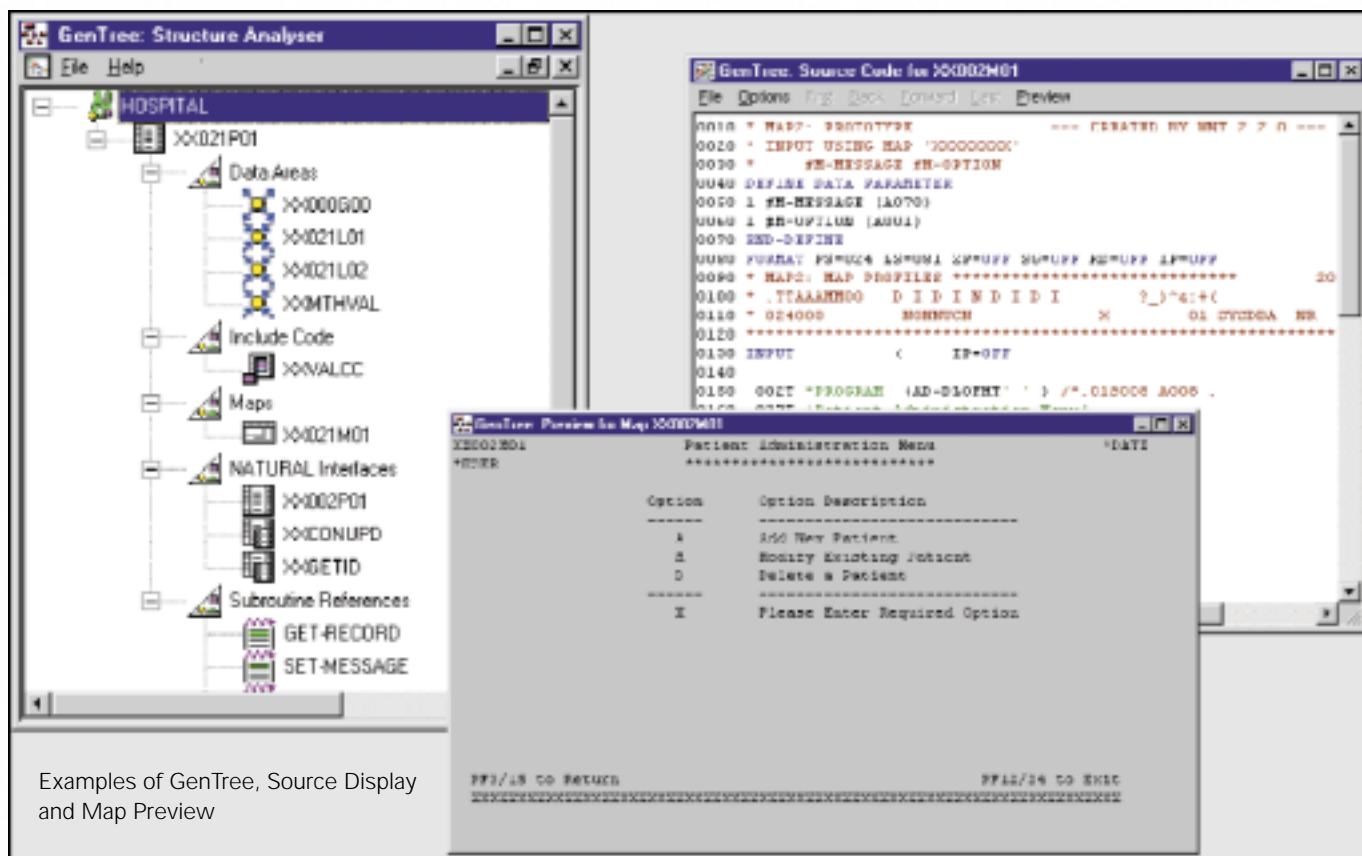
An important way in which Natural Engineer enhances maintenance productivity is its integrated structure diagramming function (known as Gen-Tree). This offers a graphical representation of Natural objects and their relationships, making it exceptionally easy to understand the structure of an application.

Icons are used to identify different types of Natural objects in a structure diagram at a glance. Where a map object forms part of the structure, a special preview function is available which, at a single mouse click, displays the map as an application end user would see it.



Example of Global Standards Definition





When working at a PC, it is easy to navigate through the structure of an application graphically and then click the mouse to bring up a window showing the source code of the selected object. The ability to switch rapidly between diagrams and the source will generate significant productivity improvements when analyzing code for required changes. And using a PC, it is easy to drill down through the application structure very rapidly, to whatever level of detail is required.

#### MORE AUTOMATED APPROACH

Natural Engineer provides powerful facilities for the automated modification of Natural source code. Changes can be made in place of, before or after identified impacts. Solutions models can easily be created in the form of TLMs (Text Logic Members). Once a solution model has been creat-

ed, and tested, the change can be applied across the whole application automatically. Where large volumes are involved, the Natural Engineer Scheduler feature can be used to apply the modifications as a batch process.

Users can also customize Natural Engineer easily to their own environment. For example, on the PC, they can view output using their browser of choice.

#### COMPLEX PROGRAM CODE

A major obstacle to maintenance productivity often not appreciated is complex program code. The complexity and effect of code in specific areas of an application can be difficult to understand and, therefore, modify. This is particularly so if application knowledge is limited, or contract development staff are being used for maintenance projects.

Natural Engineer's GenMetrics feature solves this problem by applying standard industry complexity measurements, such as Halstead or McCabe, to Natural source code. The results are presented in report form or as a series of easily understood and compared graphics. With the information GenMetrics provides, developers having relevant knowledge can concentrate on the most complex part of the application, leaving less experienced staff to handle the more routine areas of the system.

Complexity of program code can become a significant issue when application maintenance outsourcing is under consideration. Having clearly documented details of program complexity can help these companies to negotiate better contractual terms and lower costs in such situations.



# Enhanced Documentation – for Greater Clarity and Efficiency

**High-quality documentation is essential for efficient application maintenance and re-engineering. Natural Engineer gives you comprehensive system documentation that is always right up to date.**

## COMPREHENSIVE REPORTING

Another important factor in the improvement of application maintenance and re-engineering quality is the availability of better and more comprehensible system documentation. Natural Engineer provides this at a number of different levels, described below. Even at the basic level more than 30 standard reports are available with Natural Engineer.

A key benefit of Natural Engineer is that all documentation is always available online, and fully up-to-date from the repository.

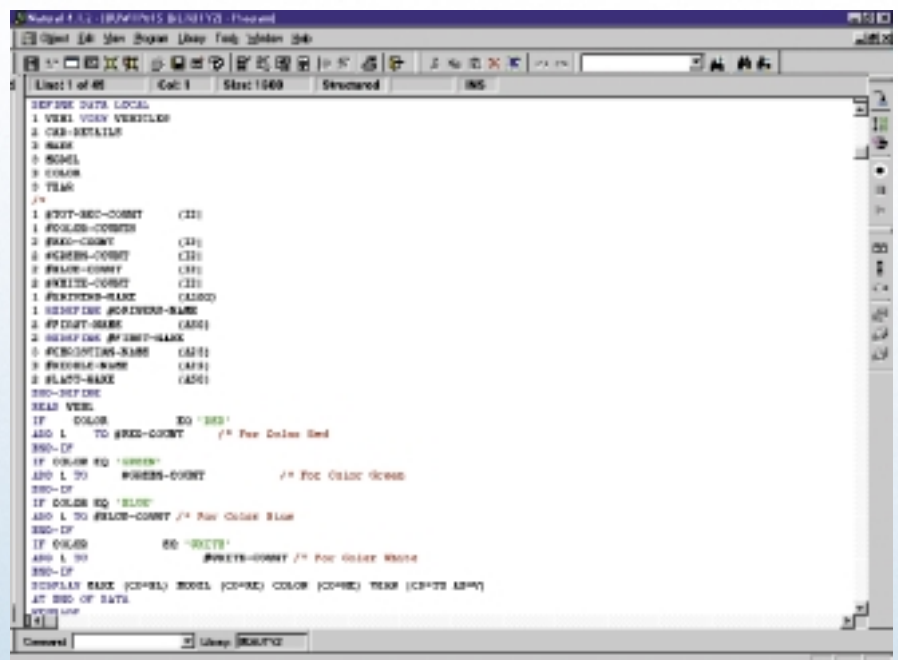
## CODE BEAUTIFICATION

An important way to improve documentation is to introduce consistency in the way source code is presented. For this purpose, Natural Engineer lets you define templates to be applied to program source code at an installation, application or individual level.

It is obvious that the older a production system is, the more complex, and frequent, the requirements are for re-engineering maintenance work. It is also very common that the development staff that created the original system are no longer available and the modification work has to be carried out by staff not familiar with the system, or even by external contractors. The consistent presentation of program source code made possible by Natural Engineer can be a considerable productivity boost in these situations.

Examples of standards that can be applied include:

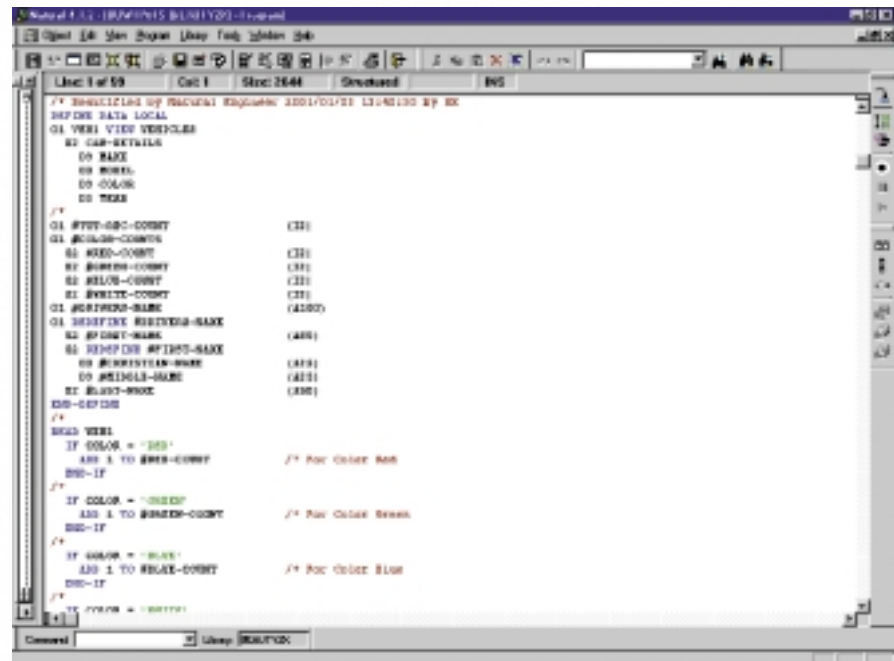
- Level numbering
- Indentation spacing
- Use of acronyms instead of symbols for logical operators and vice versa
- Statement spacing
- Line joining and splitting



```
1 VERL VERN VERLVERL
2 CIB-DETAILS
3 NAME
4 MODEL
5 COLOR
6 YEAR
7
8 1 #CIB-DETAILS-COUNT (CIB)
9 1 #CIB-DETAILS-COUNT (CIB)
10 1 #CIB-DETAILS-COUNT (CIB)
11 1 #CIB-DETAILS-COUNT (CIB)
12 1 #CIB-DETAILS-COUNT (CIB)
13 1 #CIB-DETAILS-COUNT (CIB)
14 1 #CIB-DETAILS-COUNT (CIB)
15 1 #CIB-DETAILS-COUNT (CIB)
16 1 #CIB-DETAILS-COUNT (CIB)
17 1 #CIB-DETAILS-COUNT (CIB)
18 1 #CIB-DETAILS-COUNT (CIB)
19 1 #CIB-DETAILS-COUNT (CIB)
20 1 #CIB-DETAILS-COUNT (CIB)
21 1 #CIB-DETAILS-COUNT (CIB)
22 1 #CIB-DETAILS-COUNT (CIB)
23 1 #CIB-DETAILS-COUNT (CIB)
24 1 #CIB-DETAILS-COUNT (CIB)
25 1 #CIB-DETAILS-COUNT (CIB)
26 1 #CIB-DETAILS-COUNT (CIB)
27 1 #CIB-DETAILS-COUNT (CIB)
28 1 #CIB-DETAILS-COUNT (CIB)
29 1 #CIB-DETAILS-COUNT (CIB)
30 1 #CIB-DETAILS-COUNT (CIB)
31 1 #CIB-DETAILS-COUNT (CIB)
32 1 #CIB-DETAILS-COUNT (CIB)
33 1 #CIB-DETAILS-COUNT (CIB)
34 1 #CIB-DETAILS-COUNT (CIB)
35 1 #CIB-DETAILS-COUNT (CIB)
36 1 #CIB-DETAILS-COUNT (CIB)
37 1 #CIB-DETAILS-COUNT (CIB)
38 1 #CIB-DETAILS-COUNT (CIB)
39 1 #CIB-DETAILS-COUNT (CIB)
40 1 #CIB-DETAILS-COUNT (CIB)
41 1 #CIB-DETAILS-COUNT (CIB)
42 1 #CIB-DETAILS-COUNT (CIB)
43 1 #CIB-DETAILS-COUNT (CIB)
44 1 #CIB-DETAILS-COUNT (CIB)
45 1 #CIB-DETAILS-COUNT (CIB)
46 1 #CIB-DETAILS-COUNT (CIB)
47 1 #CIB-DETAILS-COUNT (CIB)
48 1 #CIB-DETAILS-COUNT (CIB)
49 1 #CIB-DETAILS-COUNT (CIB)
50 1 #CIB-DETAILS-COUNT (CIB)
51 1 #CIB-DETAILS-COUNT (CIB)
52 1 #CIB-DETAILS-COUNT (CIB)
53 1 #CIB-DETAILS-COUNT (CIB)
54 1 #CIB-DETAILS-COUNT (CIB)
55 1 #CIB-DETAILS-COUNT (CIB)
56 1 #CIB-DETAILS-COUNT (CIB)
57 1 #CIB-DETAILS-COUNT (CIB)
58 1 #CIB-DETAILS-COUNT (CIB)
59 1 #CIB-DETAILS-COUNT (CIB)
60 1 #CIB-DETAILS-COUNT (CIB)
61 1 #CIB-DETAILS-COUNT (CIB)
62 1 #CIB-DETAILS-COUNT (CIB)
63 1 #CIB-DETAILS-COUNT (CIB)
64 1 #CIB-DETAILS-COUNT (CIB)
65 1 #CIB-DETAILS-COUNT (CIB)
66 1 #CIB-DETAILS-COUNT (CIB)
67 1 #CIB-DETAILS-COUNT (CIB)
68 1 #CIB-DETAILS-COUNT (CIB)
69 1 #CIB-DETAILS-COUNT (CIB)
70 1 #CIB-DETAILS-COUNT (CIB)
71 1 #CIB-DETAILS-COUNT (CIB)
72 1 #CIB-DETAILS-COUNT (CIB)
73 1 #CIB-DETAILS-COUNT (CIB)
74 1 #CIB-DETAILS-COUNT (CIB)
75 1 #CIB-DETAILS-COUNT (CIB)
76 1 #CIB-DETAILS-COUNT (CIB)
77 1 #CIB-DETAILS-COUNT (CIB)
78 1 #CIB-DETAILS-COUNT (CIB)
79 1 #CIB-DETAILS-COUNT (CIB)
80 1 #CIB-DETAILS-COUNT (CIB)
81 1 #CIB-DETAILS-COUNT (CIB)
82 1 #CIB-DETAILS-COUNT (CIB)
83 1 #CIB-DETAILS-COUNT (CIB)
84 1 #CIB-DETAILS-COUNT (CIB)
85 1 #CIB-DETAILS-COUNT (CIB)
86 1 #CIB-DETAILS-COUNT (CIB)
87 1 #CIB-DETAILS-COUNT (CIB)
88 1 #CIB-DETAILS-COUNT (CIB)
89 1 #CIB-DETAILS-COUNT (CIB)
90 1 #CIB-DETAILS-COUNT (CIB)
91 1 #CIB-DETAILS-COUNT (CIB)
92 1 #CIB-DETAILS-COUNT (CIB)
93 1 #CIB-DETAILS-COUNT (CIB)
94 1 #CIB-DETAILS-COUNT (CIB)
95 1 #CIB-DETAILS-COUNT (CIB)
96 1 #CIB-DETAILS-COUNT (CIB)
97 1 #CIB-DETAILS-COUNT (CIB)
98 1 #CIB-DETAILS-COUNT (CIB)
99 1 #CIB-DETAILS-COUNT (CIB)
100 1 #CIB-DETAILS-COUNT (CIB)
```

Sample Program Code

### Program Code after Beautification



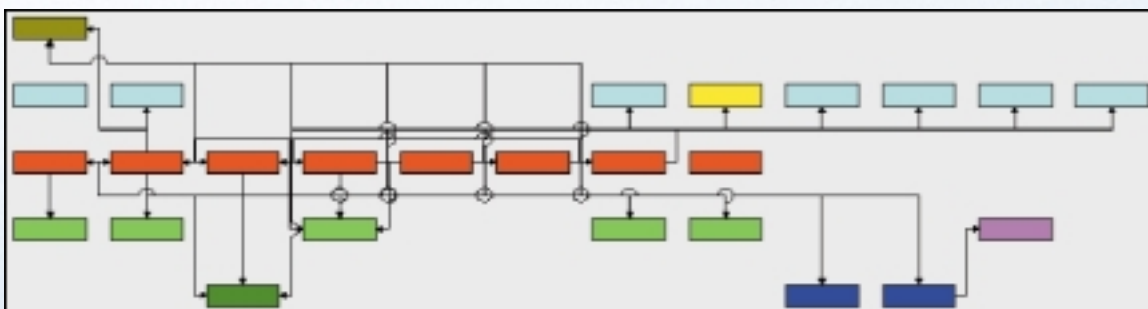
## LINKS TO THIRD-PARTY FLOW-CHARTING SOFTWARE

Natural Engineer provides an interface to popular flowcharting software packages to produce system-wide diagrams of application flow and structure.

Flow-charting packages offer a system-wide, graphical view of an application, an excellent complement to the detailed structure and code information delivered by Natural Engineer's GenTree feature.

The following products are currently supported:

- Microsoft VISIO
- Micrografx FlowCharter



## Natural Engineer interface to application flowcharting software

# Natural Version Migration – an Easier Route to New Functionality

Migrating to a different platform or to a new version of Natural may require changes to be made to applications. Natural Engineer facilitates the rapid assessment of the scope of necessary modifications and implements them quickly and efficiently.

## ASSESSING THE IMPACT OF MIGRATION

Software AG always provides detailed information on what is new, or changed, in new releases of Natural. It also highlights the changes necessary to use Natural applications on different operating platforms.

Although the differences between versions and platforms are clearly documented, it is impossible to know immediately to what extent migration will affect your application portfolio. And so it is difficult to accurately estimate the time required for the migration project, and therefore the cost. With Natural Engineer, these problems are resolved quickly and accurately.

Built into Natural Engineer are a series of search criteria reflecting changes between Natural versions. Using these as the basis of an impact analysis allows you to quickly assess the volume and complexity of changes within your particular set of applications.

## AUTOMATING THE MODIFICATION PROCESS

Having identified the necessary changes, required source modification can be applied automatically. Typically solution models in the form of Text Logic Members are created and tested. The model changes can then be applied across the entire application portfolio.

## UNIQUE DIFFERENCES

In a function-rich programming language such as Natural, there is always the possibility that unique combinations of syntax will cause new problems during a migration project. Again, once a problem has been identified, Natural Engineer can rapidly find all other similar occurrences in the application code and apply a suitable solution automatically.

## SUPPORTED MIGRATION PATHS

Currently, Natural Engineer provides support for migration from Natural Version 2.2 to Version 2.3/3.1. Other combinations of Natural versions will also be supported in the future.



# Application Restructuring – New Perspectives for Natural Systems

**Natural Engineer provides powerful, automated functions to make it easier for users to restructure applications and utilize existing processing logic in new application areas.**

## EXPLOITING THE REPOSITORY

For many re-engineering tasks today, it is not sufficient to apply individual code changes. It is increasingly necessary to restructure the existing application to enable the essential business processing logic to be used in radically different ways. Examples include creating components or objects from existing monolithic applications, layering the application to facilitate a change in the user interface or making the application ready for Web use.

The process of parsing the user's Natural source and storing the results in the repository means that Natural Engineer holds detailed information about the use of application code. This knowledge is utilized by Natural Engineer to provide automated support for application restructuring.

## BUILDING PROGRAM OBJECTS

A basic requirement for restructuring applications is to the ability to identify blocks of processing code and isolate their logic as a separate sub-program.

Natural Engineer's Object Builder feature speeds up this process significantly by automating the building of sub-programs from single Natural programs.

The illustration below shows how Natural Engineer isolates a selected section of code from a monolithic main program and creates a Natural sub-program. The CALLNAT statement is placed in the main program with an automatically created Parameter Data Area (PDA), if appropriate. All necessary data areas are also built automatically. Code that is now redundant in the main program is commented out.

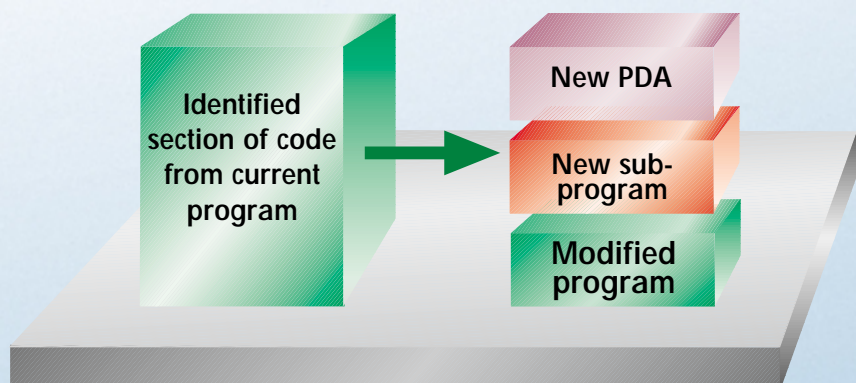
Having created the new Natural object, Natural Engineer can be used to identify other occurrences of this program code elsewhere in the application portfolio and substitute calls to the new object for the existing in-line code.

Currently, the process of selecting blocks of code to be encapsulated in a separate object is largely manual. Increasingly automated facilities for the selection process will be provided in future releases of Natural Engineer.

Once a code block has been selected for encapsulation, Natural Engineer automatically creates the object and all required linkages to the main program. This saves considerable time over manual techniques.

Restructuring is not the trivial task it might at first appear. Many factors have to be considered during the creation of each object. Described below are just two of the many logical decisions to be made when creating the new object. These examples clearly illustrate the complexity of the componentization process—complexity that Natural Engineer handles automatically.

1. Data items used in the original Natural program code outside the range of statements specified for inclusion in the new object must be added to the generated Program Data Area (PDA) associated with the subprogram. If a data item is not used outside the range of statements specified to Object Builder then the data item is not added to the PDA.
2. System variables defined inside the range of statements selected by Object Builder and used elsewhere in the program require a new data item in the PDA. Object Builder moves the system value to the data item in the new object (sub-program) being created and then changes all references in the original program code.





As can be seen from the points above, the many factors involved in restructuring decisions make it a time-consuming and error-prone task, particularly when many objects are being created from the original, single program. The use of Natural Engineer, even when only providing automated support for the generation process, can eliminate errors and significantly speed up application restructuring.

#### **STANDARD RESTRUCTURING OPTIONS**

In addition to Object Builder's basic restructuring facilities, Natural Engineer currently offers the following standard restructuring options:

- Isolation of database I/O operations (Technical Split)
- Isolation of user-presentation operations (Presentation Split)

These options are discussed in more detail below.

#### **ISOLATING DATABASE I/O OPERATIONS**

Production application code often mixes database I/O statements with processing logic. This makes it difficult to identify I/O performance problems and also to modify I/O statements that might need to change if an alternative database or file architecture is used to store the application data.

Natural Engineer's Technical Split feature provides a sophisticated solution. It automatically creates a sub-program each time I/O statements are identified in Natural source code. This results in all I/O statements being consolidated into separate objects, leaving the main program comprising only the user presentation code and the processing logic.

The Technical Split feature comes with a specific set of search criteria. These are used to locate all database I/O and related statements in the application source code, for one or more selected Data Definition Modules (DDMs). Building on the basic Object Builder technology, separate sub-programs are created containing the I/O statements, together with required data areas and processing logic, and control statements. A library of compile and runtime objects is supplied to support the storage and execution of the generated objects.

Once the technical split has been completed, it becomes much easier to interface the Natural application to new sources of data, such as a different database system or an application package. In particular, it paves the way to the use of new XML technology with existing applications—without changing the presentation and processing logic.

#### **ISOLATING USER-PRESENTATION OPERATIONS**

Another common problem with production application code is that user-presentation statements are mixed with processing logic. Isolating such statements is often highly desirable, because it makes it easy to implement a new user interface. Many production systems were designed to use an interface based on 3270 screen protocols. In the recent past there was an emphasis on providing users with a GUI interface using the PC or to support an Internet interface.

Natural Engineer's Presentation Split feature allows you to encapsulate INPUT statements and associated code into new sub-programs, and replace the code in the original program with a CALLNAT. This creates far simpler objects that can be further enhanced to support future user interfaces, including Web browsers.

The Presentation Split feature comes with a specific set of search criteria that locates all INPUT statements. The functionality then determines the exact data item requirements for each INPUT statement, which could be an inline map or an external map. It also generates a PDA that supports the transfer to the new sub-program.

All processing is automatic, with no user intervention required. This results in much higher productivity, while using a software tool to perform this task ensures accuracy, consistency, and maximum flexibility.

# Application Performance Analysis

Application performance is an increasingly important issue for Natural users. Natural Engineer provides a number of features to analyze the scope and volume of performance issues in Natural application programs.

## FINDING AND SOLVING PERFORMANCE PROBLEMS

When faced with a performance problem, Natural Engineer's powerful searching and automatic modification features can be a valuable aid in the investigation process. All occurrences of statements and constructs that could be causing a problem can easily be found and modified where required. In addition, the audit log provides an excellent tool for checking recent changes to an application system.

## MEASURING PROGRAM COMPLEXITY

The complexity of code in specific areas of program code can be difficult to understand and therefore modify. This is particularly so if application knowledge is limited, or contract development staff are being used for maintenance projects. Complex application code is also often the source of performance problems in production systems.

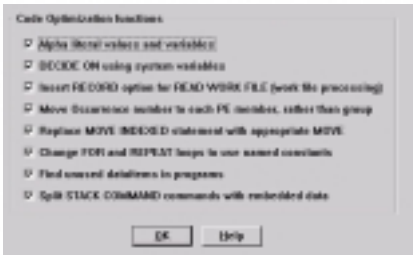
The GenMetrics feature of Natural Engineer analyzes Natural source code and measures the relative complexity of the code based on industry standard measurements, such as those defined by Halstead and McCabe. Relative complexity between Natural objects can quickly be displayed in report form, or as a series of graphics.

This means that the areas of a system to be investigated during a performance analysis project can rapidly be identified and prioritized.

## PERFORMANCE IMPACT ANALYSIS

With Natural Engineer it is possible to execute an Impact Analysis, with a specific set of supplied search criteria, to identify Natural statements and definitions that could, potentially, have a negative effect on performance. The supplied search criteria are based on the detailed information on Natural internal execution provided by Software AG.

There are currently eight performance-related functions that can be searched for (see illustration). Natural Engineer makes it easy to identify exactly where these potential problems are located in the application source code. This frees up Natural developers to spend time on deciding the benefits of implementing changes based on this analysis, not on finding the occurrences.

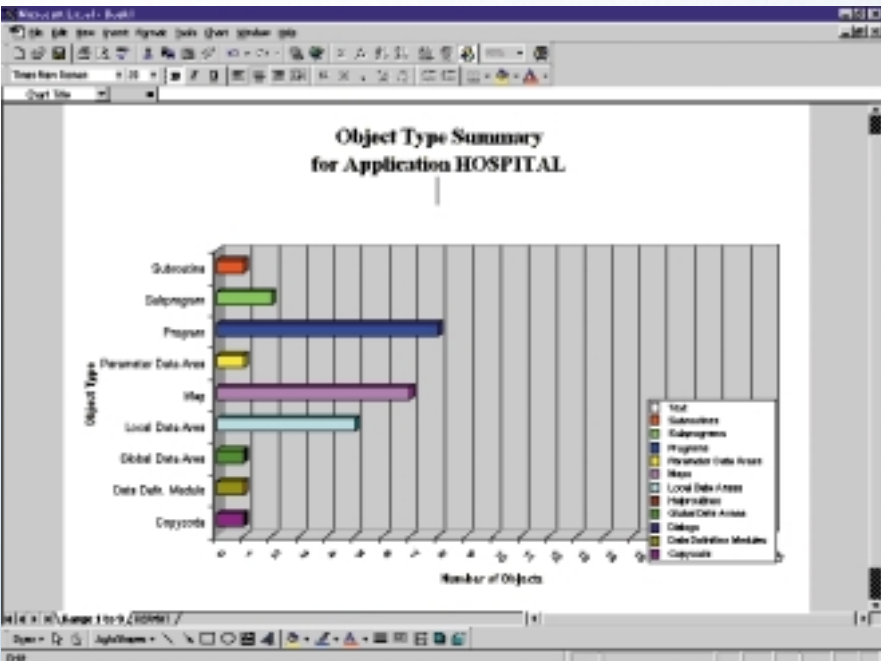


Example of Code Optimization Selection

## AUTOMATIC MODIFICATION

Having used Natural Engineer to find all occurrences of statements and constructs that could impact performance, it is possible to use the automatic modification features of the product to implement the code changes, if required. This powerful functionality should be used selectively and only when the impact of changes on application performance is clearly understood.

Application Complexity Measurement



## Additional Functions

**A number of other useful and powerful functions round out Natural Engineer's comprehensive set of features. These include support for internationalization of applications.**

### COMPARE FUNCTION

Natural Engineer provides a function that allows you to compare Natural libraries in order to identify differences in Natural source code. The Compare function can be applied to whole libraries, or to ranges of objects within libraries. It can identify differences at an object level, by timestamp or based on actual source code variations. You can view results on screen or in hard-copy form.

### INTERNATIONALIZATION SUPPORT

In an age of globalization and international company mergers and acquisitions, the ability of end users to communicate with application systems in their own, local language becomes increasingly important. This is particularly true when application systems are Web-enabled and are, therefore, being accessed, potentially, from anywhere in the world. In this scenario, revenue could be impacted by the ability of the customer or prospect to be able to use the application successfully.

Natural Engineer offers functions today to help in the developing multi-language support for existing Natural applications. A major difficulty with internationalization is the use of double-byte character sets by some languages, such as Japanese. Natural Engineer can be used to identify potential problem areas such as extra positions being required for input fields on a map.

Another potential problem arises when message texts are coded as literals within the program code. For an international implementation of the application, error messages should be stored in Natural's Message file, where multiple language versions of the message can be stored and displayed according to the user's Natural language indicator. Natural Engineer can find all occurrences of such embedded message text, making it a simple task for the user to move these to the Message file.

As companies continue to expand into the global market, Natural Engineer will provide additional functionality to facilitate the internationalization of supporting IT systems.



# Code Certification – a Critical Task

After an application has been modified, it is essential to perform comprehensive tests. Often, however, insufficient time and attention are paid to the test phase, even though correcting errors in production systems can be far more expensive than rigorous testing.

### FALSE ECONOMIES

Analysts have shown that the test phase of re-engineering projects, if conducted properly, can account for up to 50% of total project costs. However, in many projects—mainly due to time constraints—testing is cursory, and real production data is not employed. Cutting corners at this stage can have serious implications.

### TWO POWERFUL, COST-EFFECTIVE TESTING TOOLS

Natural Engineer not only helps you to modify your applications efficiently. Its two complementary products, GenAda and GenTest, enable you to test modifications thoroughly, and cost-effectively, before taking the applications back into production.

- **GenAda** makes it easy to abstract representative sets of data from production databases and sequential files. These can be used as test beds for rigorous application testing and data comparisons.
- **GenTest** fully automates the testing process and lets you define test scripts easily and flexibly. It makes it a simple task to capture the details of application execution before modification, and then run the same execution path automatically with the modified code. GenTest automatically highlights differences between the two runs.

### THE BENEFITS OF AN INTEGRATED TOOL SET

GenAda and GenTest combine with Natural Engineer to provide unique functionality. Integration between these products results in greater developer productivity and lower costs. In addition, as these tools are all based on Adabas and Natural, implementation, operating and training costs are lower when compared to alternative, standalone solutions.

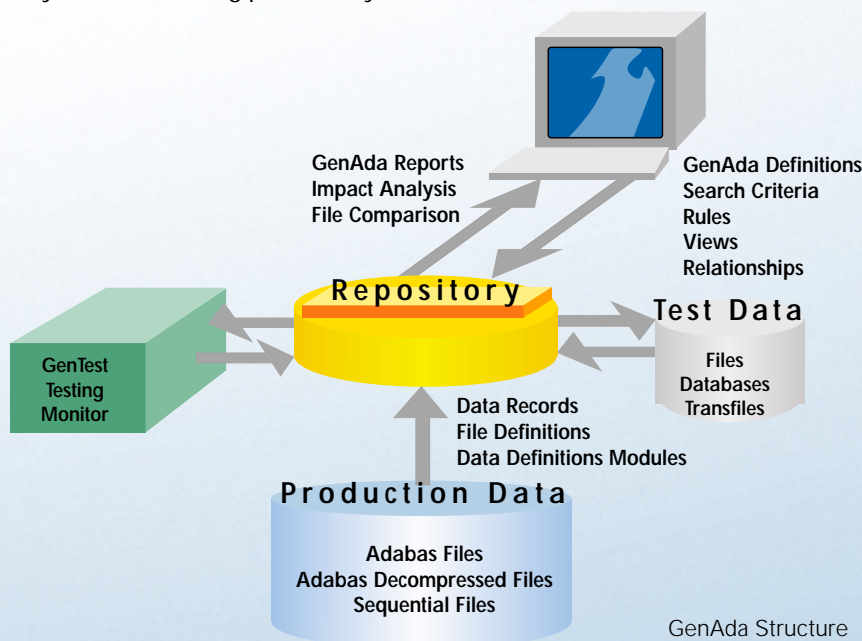
When integrated with GenTest, GenAda provides a sophisticated method of delivering representative data to the testing process. GenAda rules can be applied to fields automatically during testing.

GenTest can utilize Natural Engineer's repository to retrieve information about the structure of an application. This can be reviewed during the test process. In other words, as soon as GenTest identifies an error during testing, the user can directly access the Natural object and field involved. This greatly enhances testing productivity.

### GENADA—TESTING WITH REAL PRODUCTION DATA

Experience has shown again and again how important it is to conduct rigorous application tests with real production data. GenAda makes it easy to project an accurate image of the production data onto the testing process.

GenAda is a highly flexible tool for manipulating data in Adabas databases and sequential files. Powerful, rule-based processing with GenAda lets you rapidly develop comprehensive test databases and files. This helps you significantly reduce overall development costs, improve the turnaround for routine application maintenance, and cut the cost of re-engineering and migration projects.





GenAda includes the following features:

#### **Rule-Based Data Manipulation**

With GenAda's powerful rule-based processing, you can apply linear, semantic, euro and user-defined rules to data when creating test files and databases. A specific test scenario could require that fixed increments are to be added to date fields (linear rules) or variable increments based on specific date contexts such as end-of-quarter (semantic rules). The euro rule controls currency conversion and currency field expansion to meet EMU-compliance requirements.

GenAda even lets you link rules to individual users. For instance, you can assign national calendars or currencies to ensure that schedule test data reflects local holidays, and that currency conversion is performed correctly.

#### **Logical Relationships and Logical Views**

In real-life applications, data in multiple files are logically linked. GenAda makes it simple to define these relationships (in up to 50 files) and generate a comprehensive set of test data in a separate workfile called a Transfile. GenAda also lets you define multiple logical views for a single physical database file. This gives you maximum flexibility in defining and creating comprehensive test datasets.

#### **Database and Sequential File Support**

GenAda's comprehensive functions, including file comparison, can be applied both to Adabas files (compressed and decompressed) and

to sequential files. This gives you plenty of scope to create suitable test scenarios.

#### **Data Analysis Support**

GenAda substantially improves the productivity of database administration. It provides—in a single, easy-to-use tool—all the functions you need to find, modify and copy selected data from Adabas databases and sequential files.

#### **USING GENADA**

##### **Environment Set-Up**

GenAda extracts descriptions of data and fields from Adabas databases and sequential files, and loads these into the repository. It then retrieves data from required files for subsequent analysis by user-defined search criteria. Detailed reports allow you to monitor every stage of the process.

##### **Analysis**

GenAda provides default search criteria for specific purposes such as analyzing the impact of implementing the euro. A comprehensive set of functions is provided that enables users to define and maintain search criteria for analyzing data based on data values, formats or masks.

##### **Modification**

GenAda lets you modify data online against Adabas database files or offline against decompressed Adabas files and sequential files. Using DDMs (Data Definition Modules), you can develop test cases based on multiple views of single database files. Alternatively, the GenAda Transfile feature enables you to combine data that is logically linked in up to 50 files to create comprehensive test beds.

#### **GENTEST—AUTOMATING THE TESTING PROCESS**

GenTest is a powerful testing workbench that automates the testing process. This reduces the manual effort required and speeds up the overall test cycle. GenTest's powerful test automation facilities simplify application quality assurance following routine maintenance or an application re-engineering process.

The use of an automated tool permits more comprehensive testing of application logic. The result is lower total costs for re-engineering and maintenance projects, and improved quality and integrity of application code.

GenTest offers a number of significant features:

##### **Flexible Script Management**

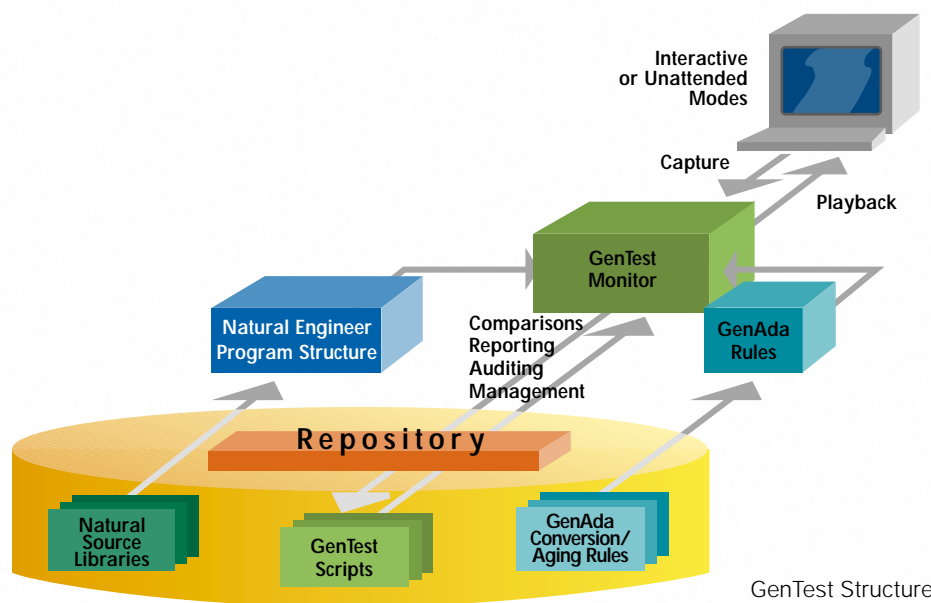
With GenTest it is possible to generate new test scripts, re-execute existing tests, manipulate scripts to reflect new test requirements, and compare the results of multiple test executions. A single test can also be replicated automatically against multiple data values for an input field on a screen, reducing the need for multiple scripts and test executions.

##### **Interactive Test Review**

GenTest's online, interactive test review process facilitates the management and documentation of scripts for future application development and maintenance tasks.

##### **Reviewing Application Structure**

GenTest can utilize the Natural Engineer repository to retrieve information about the structure of an appli-



GenTest Structure

cation. This can be reviewed during the testing process so that when GenTest identifies an error, you can directly, interactively access the Natural object and field involved.

#### Auditing

Comprehensive reporting functions provide full audit information covering all stages of testing. This allows you to fully monitor the testing process and associated costs.

#### Unattended Operation

GenTest supports fully unattended operation, letting you schedule testing at convenient times, such as overnight. A comprehensive scheduling function is integrated with GenTest to control all aspects of unattended processing.

#### USING GENTEST

GenTest captures online screen data from an application test execution, analyzes the test script and builds a repository of information. You can review the test script, see the results of the data input on the application, and compare the differences between multiple GenTest executions of the same script. Once generated, you can manipulate test scripts by changing the data input, changing the order of test processes, automatically replicating the script multiple times, and re-executing and auditing tests.

#### Initial Script Creation

The first step in using GenTest is to define and execute the application test script. This provides GenTest's base data. The data streams are captured, analyzed and fed into the automatic script process. For Natural applications, details of objects, maps and fields are included.

GenTest extracts the captured output and loads it into the repository. The tool identifies data streams of screen inputs and outputs, and builds the test script.

#### Script Management and Comparison

The tool provides full control facilities to modify and/or automatically re-run captured GenTest scripts.

A previously captured test can be re-executed for automatic identification of the differences between two test runs. These differences are highlighted for interactive screen review, enabling you to verify the results of the tests. Differences can be marked for change or accepted as valid.

#### Management Control

GenTest facilitates overall management control of the test phase, helping to avoid cost overruns and late delivery. You can review testing coverage at any time to identify which objects have been tested by a script, and which test processes have been applied and completed.

## The Future

**Natural Engineer is designed to use a consistent architecture and Software AG will continue to deliver new functions to boost the productivity of ongoing Natural use for mission-critical applications.**

### LEVERAGING REPOSITORY INFORMATION

The fully-parsed Natural source code stored in the Natural Engineer repository represents a single point of reference about the nature and functionality contained in an enterprise's application portfolio. Software AG will continue to explore new ways of leveraging this valuable information to enable customers to continue to reduce the overall costs of maintaining application systems built with Natural, and also to utilize the investment in production Natural systems in new, cost-effective application areas.

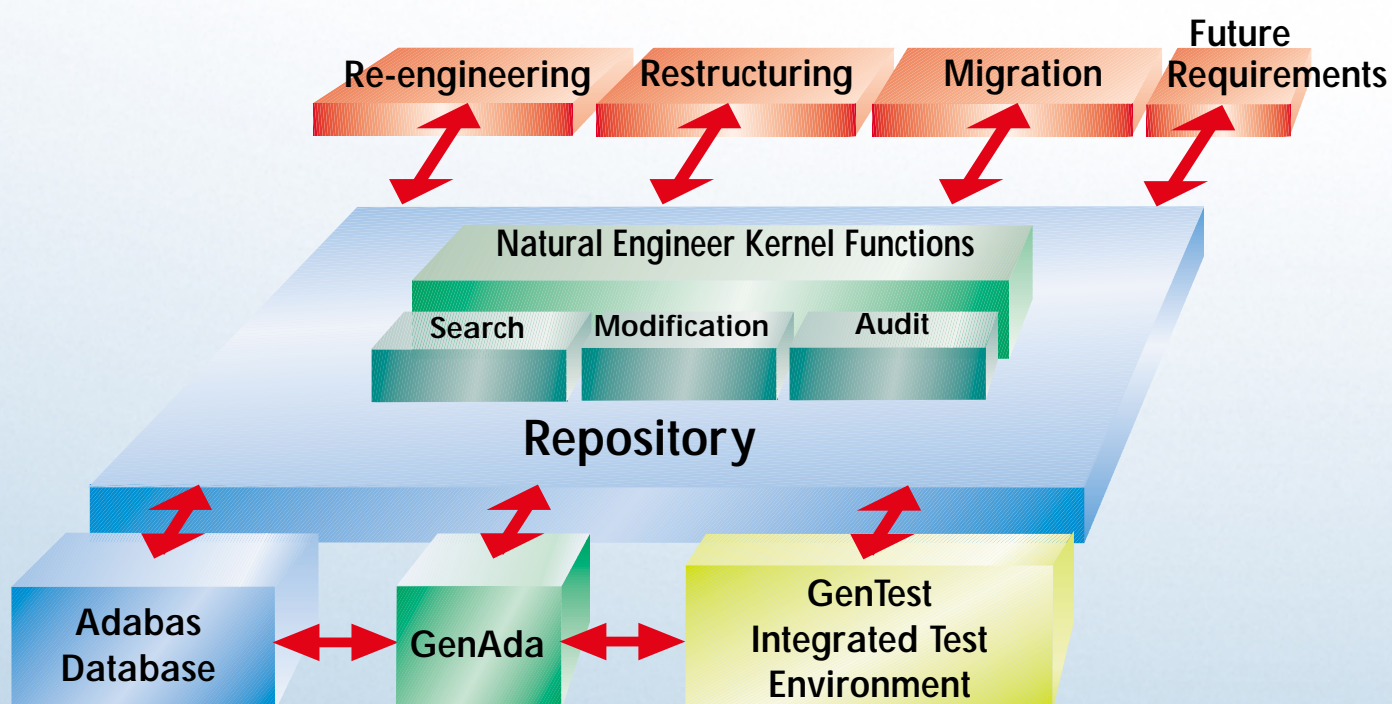
### A CONSISTENT ARCHITECTURE

All functions planned for the future are also based on the common architecture, using the Natural Engineer repository as a single source of information on the application source code used in the enterprise.

The testing tools, GenAda and GenTest form an integral part of Software AG's end-to-end solution for re-engineering and maintenance productivity. As such, both products will continue to be developed to provide appropriate complementary functionality for new versions of Natural Engineer.

The enhancements already in development include the following functions.

Integrated Architecture for Future Development





## WEB-ENABLING NATURAL APPLICATIONS

An important planned feature of Natural Engineer is the ability to automate the process of Web-enabling Natural applications. Natural Engineer's Object Builder HTML function will restructure applications and create new objects—reusing the presentation and processing logic, and generating HTML within the framework of the Natural Web Interface. The converted applications will then execute directly on the Web.

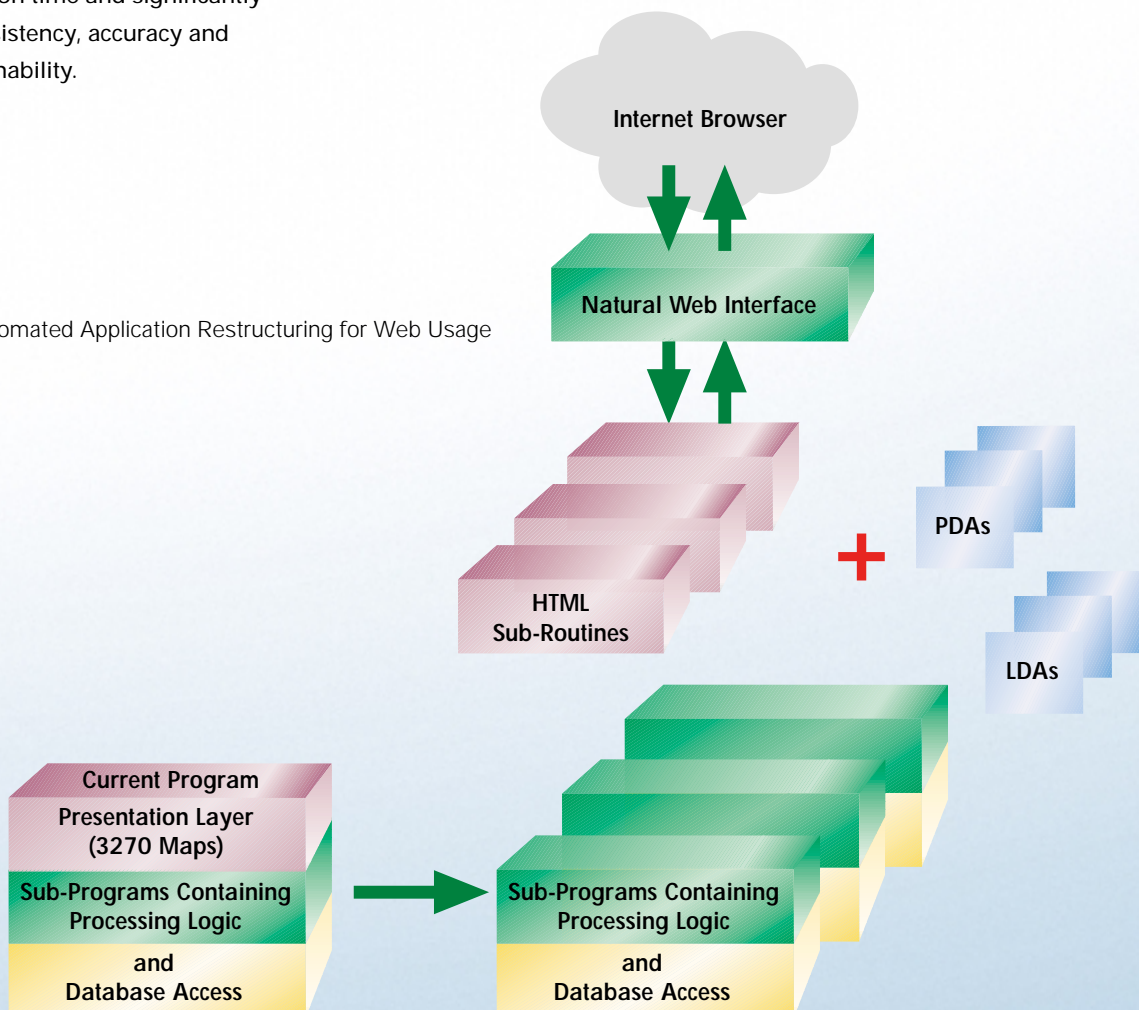
By completely automating this very complex and error-prone task, Natural Engineer will significantly reduce implementation time and significantly improve consistency, accuracy and code maintainability.

The diagram below illustrates the process in a simplified form. It shows the restructuring of the application from an existing Natural object with a Screen Map to a set of objects using HTML presentation.

For each Map encountered in the application, Natural Engineer will create an equivalent sub-routine using HTML plus relevant Natural Web Interface commands. A sub-routine is also created to handle the processing logic and rules that may be associated with the original Map.

All necessary data areas and links to the original Natural program are also generated automatically, fully resolving the complex navigational and processing issues associated with Web-enablement. The resulting restructured application can be accessed immediately via the Internet.

Automated Application Restructuring for Web Usage





## TECHNICAL ENVIRONMENT

This document is based on the  
Natural Engineer 4.2.1 Base Version.

The minimum operating environment to support Natural Engineer Version 4.2.1 is:

<b>Mainframe:</b>	Adabas Version 6.2
	Natural Version 2.3.4
	Predict Version 3 (optional)
<b>PC: Windows NT</b>	Natural Version 4.1.2.1
	Adabas Version 3.1.1
	Internet Explorer Version 3 or above (optional)
	MicroGrafx Flowcharter Version 7 (optional)
	Microsoft VISIO Version 4 (optional)

Software AG and/or all Software AG products are either trademarks or registered trademarks of Software AG. Other product and company names mentioned herein may be the trademarks of their respective owners.

**Software AG**  
**Corporate Headquarters**  
Uhlandstraße 12  
64297 Darmstadt  
Germany  
Tel.: +49-61 51-92-0  
Fax: +49-61 51-92-11 91  
[www.softwareag.com](http://www.softwareag.com)

NAT/SB03E0101 003

